

A Formalization of the Development Closedness Criterion for Left-Linear Term Rewrite Systems

Christina Kohl

Department of Computer Science
University of Innsbruck
Innsbruck, Austria
christina.kohl@uibk.ac.at

Aart Middeldorp

Department of Computer Science
University of Innsbruck
Innsbruck, Austria
aart.middeldorp@uibk.ac.at

Abstract

Several critical pair criteria are known that guarantee confluence of left-linear term rewrite systems. The correctness of most of these have been formalized in a proof assistant. An important exception has been the development closedness criterion of van Oostrom. Its proof requires a high level of understanding about overlapping redexes and descendants as well as several intermediate results related to these concepts. We present a formalization in the proof assistant Isabelle/HOL. The result has been integrated into the certifier CēTA.

CCS Concepts: • Theory of computation → Equational logic and rewriting; Logic and verification.

Keywords: formalization, term rewriting, confluence

ACM Reference Format:

Christina Kohl and Aart Middeldorp. 2023. A Formalization of the Development Closedness Criterion for Left-Linear Term Rewrite Systems. In *Proceedings of the 12th ACM SIGPLAN International Conference on Certified Programs and Proofs (CPP '23), January 16–17, 2023, Boston, MA, USA*. ACM, New York, NY, USA, 14 pages. <https://doi.org/10.1145/3573105.3575667>

1 Introduction

Rewriting is a pervasive concept in mathematics, computer science, and other areas; simplification of expressions constitutes rewriting, the execution of a program can be seen as a rewrite sequence on program states, and in fact probably almost any development according to a set of fixed rules can be considered rewriting. In *term rewriting*, we assume that the objects which are rewritten are terms. This yields a powerful formalism which is crucial for simplification in automated theorem proving, it provides tools to analyze security protocols, but it is also a versatile method in program

verification, to name only a few application areas. In fact, term rewriting is a Turing-complete model of computation, and provides methods to investigate important properties of computation and simplification processes on an abstract level.

A term rewrite system \mathcal{R} is a set of directed equations, so-called rewrite rules, which induces a relation $\rightarrow_{\mathcal{R}}$ on terms. Besides termination, which forbids infinite computations, *confluence* has been conceived as one of the central properties of rewriting. A rewrite system \mathcal{R} is confluent if for all terms s, t and u such that $s \rightarrow_{\mathcal{R}}^* t$ and $s \rightarrow_{\mathcal{R}}^* u$ (here $\rightarrow_{\mathcal{R}}^*$ denotes the transitive reflexive closure of $\rightarrow_{\mathcal{R}}$) there exists a term v such that $t \rightarrow_{\mathcal{R}}^* v$ and $u \rightarrow_{\mathcal{R}}^* v$ (see Figure 2). Confluence is equivalent to the Church–Rosser property, introduced in 1936 by Church and Rosser [6] to show the consistency of the λ I-calculus, and guarantees that normal forms (which are terms t such that $t \rightarrow_{\mathcal{R}} u$ for no term u) are unique.

Although undecidable in general, several sufficient conditions for confluence are known. The best-known ones are based on restricting the way in which critical pairs can be joined. These apply to left-linear rewrite systems and are covered in textbooks on term rewriting [5, 27]. The emergence of tools that aim to prove confluence automatically and compete in the Confluence Competition¹ (CoCo) [15] has led to many new techniques (e.g. [1, 3, 9, 11, 16]) also for rewrite systems that are not left-linear. Software tools may contain bugs and confluence tools are no exception. Indeed, YES/NO conflicts (i.e., instances where tools yield contradictory answers) are occasionally observed in CoCo, partly explaining the interest in certifying the output of confluence tools. An important first step is to formalize the techniques used in confluence tools in a proof assistant.

Starting from the fundamental result by Knuth and Bendix [13], stating that a terminating rewrite system is confluent if and only if all its critical pairs are joinable, several confluence results have been formalized in the recent past. An overview can be found in Section 3.

A well-known condition that has been considered as a valuable addition to ongoing formalization efforts [19, 23], but has never been formalized so far, is the result by van Oostrom [30] that a left-linear rewrite system is confluent if its critical pairs are development closed. In [10] it is suggested

¹<http://project-coco.uibk.ac.at/>



This work is licensed under a Creative Commons Attribution 4.0 International License.

CPP '23, January 16–17, 2023, Boston, MA, USA
© 2023 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0026-2/23/01.
<https://doi.org/10.1145/3573105.3575667>

to use proof terms [27, Chapter 8] to obtain a rigorous proof. In [19] it is further suggested that a formalization of residual theory might be helpful. Following these suggestions we describe in this paper our formalization of the development closedness criterion in the proof assistant Isabelle/HOL [22].

In the following we first recall basic notions and results on term rewriting and provide some preliminary information about the Isabelle formalization. Then we give an overview of related work (Section 3). In Section 4 we introduce development closed critical pairs, state the corresponding confluence theorem and give an overview of the structure of its proof. In the following sections we provide details about proof terms (Section 5) and unification of linear and variable disjoint terms (Section 6). Both of these concepts play important roles in the formalization. In Section 7 we finally present the full proof of the development closedness theorem and in Section 8 we provide some details on the Isabelle formalization. Integration of this result into the verifier CeTA is described in Section 9. In Section 10 we summarize our contributions and provide an outlook on future work.

2 Preliminaries

Previous experience with Isabelle is not necessary for following our proof, but familiarity with the basics of term rewriting [5, 27] will be helpful. Below we recall some definitions and notations.

2.1 Term Rewriting

Let \mathcal{F} be a signature and \mathcal{V} a set of variables disjoint from \mathcal{F} . By $\mathcal{T}(\mathcal{F}, \mathcal{V})$ we denote the set of terms over \mathcal{F} and \mathcal{V} . A context is a term containing exactly one occurrence of the special constant symbol \square , which is called hole. If C is a context then $C[t]$ denotes the result of replacing the hole by t . Positions are strings of positive natural numbers used to address subterm occurrences. The set of positions of a term t is defined as $\mathcal{Pos}(t) = \{\epsilon\}$ if t is a variable and as $\mathcal{Pos}(t) = \{\epsilon\} \cup \{iq \mid 1 \leq i \leq n \text{ and } q \in \mathcal{Pos}(t_i)\}$ if $t = f(t_1, \dots, t_n)$. The subterm of t at position $p \in \mathcal{Pos}(t)$ is defined as $t|_p = t$ if $p = \epsilon$ and as $t|_p = t_i|_q$ if $p = iq$ and $t = f(t_1, \dots, t_n)$. We write $s[t]_p$ for the result of replacing the subterm at position p of s with t . The symbol in t at position $p \in \mathcal{Pos}(t)$ is denoted by $t(p)$. We write $q \leq p$ if $qq' = p$ for some position q' , in which case $p \setminus q$ is defined to be q' . Furthermore $q < p$ if $q \leq p$ and $q \neq p$. Finally, positions q and p are parallel, written as $q \parallel p$, if neither $q \leq p$ nor $p < q$. We denote the subset of $\mathcal{Pos}(t)$ of non-variable positions (i.e., the positions $p \in \mathcal{Pos}(t)$ such that $t|_p \notin \mathcal{V}$) by $\mathcal{Pos}_{\mathcal{F}}(t)$. We write $\mathcal{Var}(t)$ for the set of variables occurring in the term t . A term is linear if every variable occurs at most once in it. Given a linear term t , we write $\text{var}(t)$ for the list (x_1, \dots, x_n) of variables appearing in t in some fixed order. Moreover, $\text{vpos}(t)$ denotes the corresponding list (p_1, \dots, p_n) of positions in t where these variables occur. A substitution is a map σ from \mathcal{V} to

$\mathcal{T}(\mathcal{F}, \mathcal{V})$ such that its domain $\{x \in \mathcal{V} \mid \sigma(x) \neq x\}$ is finite. We write $t\sigma$ for the result of applying σ to the term t .

A rewrite rule is a pair of terms (ℓ, r) , written $\ell \rightarrow r$. A rewrite rule $\ell \rightarrow r$ is left-linear if ℓ is linear. A variant of a rewrite rule is obtained by renaming its variables. A term rewrite system (TRS) is a set of rewrite rules over a signature. In the sequel, signatures are left implicit. A TRS is *left-linear* if all its rules are left-linear. A TRS \mathcal{R} induces the relation $\rightarrow_{\mathcal{R}}$ defined on terms as follows: $s \rightarrow_{\mathcal{R}} t$ if there exists a position $p \in \mathcal{Pos}(s)$, a rewrite rule $\ell \rightarrow r \in \mathcal{R}$ and a substitution σ such that $s|_p = \ell\sigma$ and $t = s[r\sigma]_p$. The *multi-step* relation $\twoheadrightarrow_{\mathcal{R}}$ is inductively defined on terms as follows:

- $x \twoheadrightarrow_{\mathcal{R}} x$ for all variables x ,
- $f(s_1, \dots, s_n) \twoheadrightarrow_{\mathcal{R}} f(t_1, \dots, t_n)$ if $s_i \twoheadrightarrow_{\mathcal{R}} t_i$ for all $1 \leq i \leq n$, and
- $\ell\sigma \twoheadrightarrow_{\mathcal{R}} r\tau$ if $\ell \rightarrow r \in \mathcal{R}$ and $\sigma(x) \twoheadrightarrow_{\mathcal{R}} \tau(x)$ for all $x \in \mathcal{Var}(\ell)$.

From the definition it easily follows that $\rightarrow_{\mathcal{R}} \subseteq \twoheadrightarrow_{\mathcal{R}}$ for any TRS \mathcal{R} . Whenever the underlying TRS \mathcal{R} is clear from the context, we omit the index in $\twoheadrightarrow_{\mathcal{R}}$ and simply write \twoheadrightarrow .

The TRS of the following example will reappear several times throughout the paper to illustrate important concepts.

Example 2.1. Consider the left-linear TRS \mathcal{R}_1 consisting of the rules:

$$\begin{array}{ll} \alpha & h(f(x, g(y))) \rightarrow h(f(x, g(x))) & \gamma & g(a) \rightarrow g(b) \\ \beta & f(g(x), y) \rightarrow f(g(x), g(x)) & \delta & b \rightarrow a \end{array}$$

There exists a rewrite step

$$h(f(g(a), g(a))) \rightarrow h(f(g(a), g(a)))$$

using β at position 1. There exists a multi-step

$$h(f(g(a), g(a))) \twoheadrightarrow h(f(g(b), g(g(b))))$$

This can be seen by applying the third case of the definition and using rule α . Then the substitution σ must contain $x \mapsto g(a)$ while the substitution τ contains $x \mapsto g(b)$. The definition of multi-step requires that $g(a) \twoheadrightarrow g(b)$ which can be achieved by applying rule γ .²

A *critical overlap* $(\ell_1 \rightarrow r_1, p, \ell_2 \rightarrow r_2)_{\sigma}$ of a TRS \mathcal{R} consists of variants $\ell_1 \rightarrow r_1$ and $\ell_2 \rightarrow r_2$ of rewrite rules in \mathcal{R}_1 without common variables, a position $p \in \mathcal{Pos}_{\mathcal{F}}(\ell_2)$, and a most general unifier σ of ℓ_1 and $\ell_2|_p$. From a critical overlap $(\ell_1 \rightarrow r_1, p, \ell_2 \rightarrow r_2)_{\sigma}$ we obtain a critical peak

$$\ell_2\sigma[r_1\sigma] \leftarrow \ell_2\sigma[\ell_1\sigma]_p = \ell_2\sigma \rightarrow r_2\sigma$$

and the corresponding *critical pair* $\ell_2\sigma[r_1\sigma] \approx r_2\sigma$. A critical pair $s \approx t$ is *joinable* if $s \rightarrow_{\mathcal{R}}^* u$ and $t \rightarrow_{\mathcal{R}}^* u$ for some term u . A left-linear TRS without critical pairs is called *orthogonal*.

²The substitution for the variable y does not matter since y does not appear in the right-hand side of α

Example 2.2. The TRS \mathcal{R}_1 of Example 2.1 has one critical overlap between α and β and one between α and γ as well as β and γ . Hence we obtain the critical peaks displayed in Figure 1. There the arrows are labeled by the corresponding rewrite rules and positions. Note that on the right only steps at the root position ϵ are allowed.

A relation \rightarrow has the *diamond property* if $\leftarrow \cdot \rightarrow \subseteq \rightarrow \cdot \leftarrow$ (Figure 2). The following well-known result [5, Chapter 2] connects the diamond property with confluence.

Lemma 2.3. *Let \rightarrow , \rightarrow_1 and \rightarrow_2 be binary relations.*


1. *If \rightarrow has the diamond property then \rightarrow is confluent.*
2. *If $\rightarrow_1 \subseteq \rightarrow_2 \subseteq \rightarrow_1^*$ and \rightarrow_2 is confluent then \rightarrow_1 is confluent.*

When applying this lemma to prove that development closed critical pairs imply confluence for left-linear TRSs, we will first instantiate \rightarrow in the first item with \Rightarrow to obtain confluence of \Rightarrow . Then we can use the second item with the property $\rightarrow \subseteq \Rightarrow \subseteq \rightarrow^*$ to establish confluence of \rightarrow .

2.2 Isabelle/HOL

Our formalization is developed in the proof assistant Isabelle/HOL [22] and is based on the Isabelle Formalization of Rewriting (IsaFoR),³ a library of formalized results related to term rewriting. Our contribution relies on the existing formalizations of unification and critical pairs described in [25]. The Isabelle theory files as well as HTML versions of the theories can be found at

<http://informatik-protem.uibk.ac.at/cpp2023/>

The HTML versions provide a nice way to view Isabelle code without having to install Isabelle. In addition we annotated important results in this paper by a -symbol which directly links to the HTML presentation of the corresponding result.

3 Related Work

For terminating rewrite systems, as mentioned in the introduction, confluence amounts to checking the joinability of critical pairs. This landmark result by Knuth and Bendix [13] has been formalized in ACL2 [24], in PVS [8] and in Isabelle/HOL [25]. For non-terminating rewrite systems, orthogonality is the simplest sufficient criterion for confluence. It has been formalized in Isabelle/HOL [20] and in PVS [23]. The former contains the extension to weak orthogonality in which trivial critical pairs are allowed. It also describes the formalization of sufficient conditions for non-joinability of critical pairs based on unification, discrimination pairs [2], interpretations, and tree automata [7]. Powerful transformation techniques for (non-)confluence based on redundant rules [16] and rule labeling [21] have also been formalized in Isabelle/HOL. The direct predecessor of our work is [19], in

³<http://cl-informatik.uibk.ac.at/isafor>


which the classical critical pair criteria based on strong confluence and parallel rewriting of Huet [12] and Toyama [28] have been formalized.

4 Development Closed Critical Pairs

Here we restate the confluence criterion by van Oostrom [29] and provide a high-level overview of the proof. Shortened versions of the same proof can be found in [30] and [27, Chapter 11].

Definition 4.1. A TRS \mathcal{R} is development closed if for every critical pair $s \approx t$ of \mathcal{R} we have $s \Rightarrow_{\mathcal{R}} t$.

The development closedness criterion first appeared in [29], where it was introduced for the larger class of *higher-order* pattern rewrite systems. The earlier results of Huet [12] and Toyama [28] are restricted to first-order rewrite systems due to the use of parallel rewriting. We formalized the development closedness criterion for (first-order) TRSs.

Theorem 4.2. *If a TRS \mathcal{R} is left-linear and development closed then $\Rightarrow_{\mathcal{R}}$ has the diamond property.* 

As mentioned before, the diamond property of $\Rightarrow_{\mathcal{R}}$ immediately yields confluence of the TRS \mathcal{R} by Lemma 2.3.

Corollary 4.3. *If a TRS \mathcal{R} is left-linear and development closed then it is confluent.*

The TRS of the following example will reappear several times throughout the paper to illustrate important concepts.

Example 4.4. Consider again the left-linear TRS \mathcal{R}_1 of Example 2.1. One easily verifies that all three critical pairs of \mathcal{R}_1 computed in Example 2.2 are development closed. Hence the TRS \mathcal{R}_1 is confluent.

When formalizing Theorem 4.2 we could essentially follow the proof steps described by van Oostrom in [29, 30]. The main differences lie in the representation of rewrite steps as proof terms and the explicit construction of the intermediate contexts, substitutions, and rewrite steps used in the proof. In the following we outline the proof steps. We start by looking at two arbitrary multi-steps $t \Leftarrow s \Rightarrow u$ and need to show that there exist multi-steps $t \Rightarrow v \Leftarrow u$ for some term v . To facilitate formalization we model these multi-steps as proof terms, i.e., first-order terms over an extended signature, and then use operations on these proof terms, like residuals and joins [27], to show how the required multi-steps can be constructed. Let A denote the proof term witnessing $s \Rightarrow t$ and B denote the proof term witnessing $s \Rightarrow u$. The goal is to construct proof terms witnessing $t \Rightarrow v$ and $u \Rightarrow v$ for some term v . We proceed by well-founded induction on the amount of overlap between the proof terms A and B . The case where A and B do not overlap is straightforward since taking the residuals A / B and B / A immediately yields the desired result. A proof sketch of the overlapping case is depicted in Figure 3, which goes back to van Oostrom [29, 30]. Here

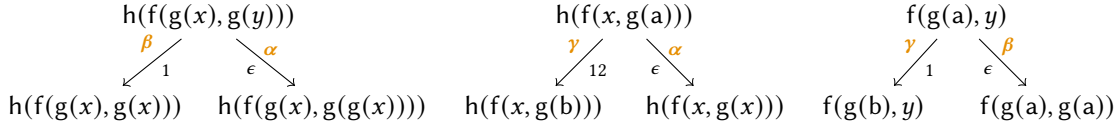


Figure 1. The critical peaks of \mathcal{R}_1 .

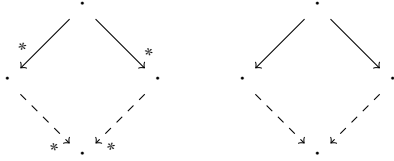


Figure 2. Confluence and the diamond property.

terms are represented as triangles. The redexes contracted by A are marked as red triangles inside the term s , the redexes contracted by B are marked as green triangles. In the picture the red and green triangles overlap at several positions. In such a case we can select an innermost overlap consisting of a step $s \rightarrow t'$ witnessed by proof term Δ_1 and a step $s \rightarrow u'$ witnessed by Δ_2 . In the picture this is indicated by the small shaded red and green triangles. We then use the fact that the TRS is development closed to close this overlap with some proof term D , represented by the solid black triangles in the picture. Using D and some additional operations on proof terms we can construct the proof terms A / Δ_1 and $D \sqcup (B - \Delta_2) / \Delta_1$ witnessing $t' \rightarrow t$ and $t' \rightarrow u$ respectively. Finally we need to show that the overlap between these new proof terms is less than between A and B to be able to apply the induction hypothesis.

The key ingredients for the proof are operations on proof terms and their properties as well as measuring the amount of overlap and the notion of an innermost overlap between A and B . Another important ingredient, which is missing from [29, 30], is the unification of linear and variable disjoint terms. This is used to obtain the closing step D for the innermost overlap. All of these concepts will be covered in detail in the next sections. We then return to the proof of Theorem 4.2 in Section 7.

5 Proof Terms

Proof terms represent computations in term rewriting. They were introduced by van Oostrom and de Vrijer for first-order left-linear rewrite systems to study equivalence of reductions in [31] and [27, Chapter 8].

Proof terms are built from function symbols, variables, and rule symbols. Rule symbols represent rewrite rules and have a fixed arity which is the number of different variables in the represented rule. In this way we can represent any multi-step as a proof term. The special case of a proof term

with only one rule symbol corresponds to a single step and a proof term without any rule symbols denotes an empty step.

We use Greek letters ($\alpha, \beta, \gamma, \dots$) for rule symbols and uppercase letters (A, B, C, \dots) for proof terms. If α is a rule symbol then $\text{lhs}(\alpha)$ ($\text{rhs}(\alpha)$) denotes the left-hand (right-hand) side of the rewrite rule denoted by α . Furthermore $\text{var}(\alpha) = \text{var}(\text{lhs}(\alpha))$ and similarly $\text{vpos}(\alpha) = \text{vpos}(\text{lhs}(\alpha))$. The length of this list is the arity of α . Given a rule symbol α with $\text{var}(\alpha) = (x_1, \dots, x_n)$ and terms t_1, \dots, t_n , we write $\langle t_1, \dots, t_n \rangle_\alpha$ for the substitution $\{x_i \mapsto t_i \mid 1 \leq i \leq n\}$. Given a proof term A , its source $\text{src}(A)$ and target $\text{tgt}(A)$ are computed by the following clauses:

$$\begin{aligned} \text{src}(x) &= \text{tgt}(x) = x \\ \text{src}(f(A_1, \dots, A_n)) &= f(\text{src}(A_1), \dots, \text{src}(A_n)) \\ \text{src}(\alpha(A_1, \dots, A_n)) &= \text{lhs}(\alpha)\langle \text{src}(A_1), \dots, \text{src}(A_n) \rangle_\alpha \\ \text{tgt}(f(A_1, \dots, A_n)) &= f(\text{tgt}(A_1), \dots, \text{tgt}(A_n)) \\ \text{tgt}(\alpha(A_1, \dots, A_n)) &= \text{rhs}(\alpha)\langle \text{tgt}(A_1), \dots, \text{tgt}(A_n) \rangle_\alpha \end{aligned}$$

The proof term A is a witness of the multi-step $\text{src}(A) \rightarrow^* \text{tgt}(A)$. For every multi-step there exists a proof term witnessing it. Proof terms A and B are said to be *co-initial* if they have the same source. The following example illustrates these concepts.

Example 5.1. Consider again the TRS \mathcal{R}_1 of Example 2.1 and the proof terms

$$A = \alpha(\gamma, a) \quad B = h(\beta(a, \gamma))$$

A and B represent the multi-steps

$$\begin{aligned} \text{src}(A) &= h(f(g(a), g(a))) \rightarrow^* h(f(g(b), g(g(b)))) = \text{tgt}(A) \\ \text{src}(B) &= h(f(g(a), g(a))) \rightarrow^* h(f(g(a), g(a))) = \text{tgt}(B) \end{aligned}$$

The proof terms A and B are co-initial since they have the same source.

Example 5.2. The critical peaks in Figure 1 are closed by the proof terms

$$\alpha(g(x), x) \quad \alpha(x, b) \quad \beta(\delta, y)$$

confirming the development closedness of the TRS \mathcal{R}_1 of Example 2.1.

In the setting of left-linear TRSs we can extend the definition of src to contexts of proof terms by adding the clause $\text{src}(\square) = \square$. Doing the same for tgt or for arbitrary TRSs however could lead to more than one hole appearing in the computation. The following result is an easy consequence of the idempotence of src and tgt .

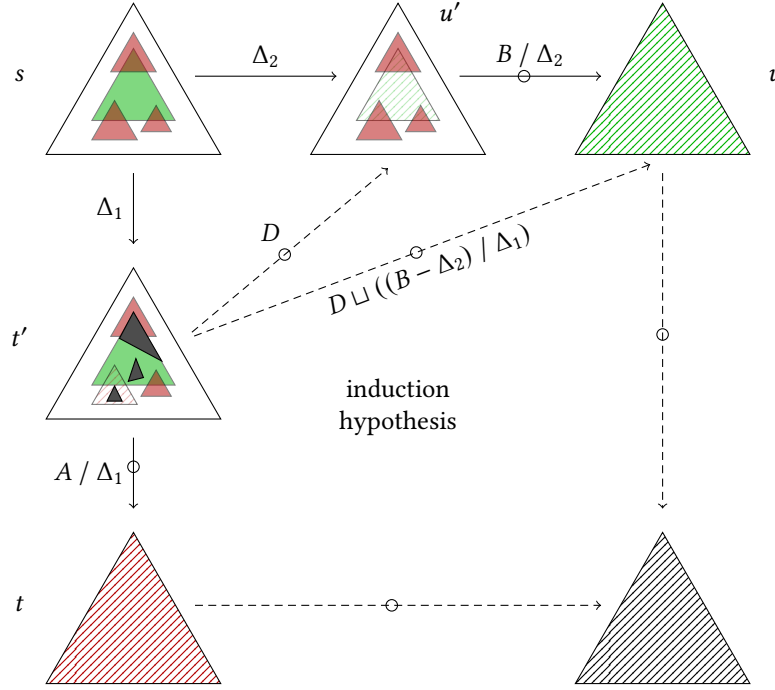


Figure 3. Picture for the step case in the proof of Theorem 4.2 (with proof terms).

Lemma 5.3. For any substitution σ , proof term context C , and proof term A we have

$$\begin{aligned} \text{src}(A\sigma) &= \text{src}(\text{src}(A)\sigma) \\ \text{tgt}(A\sigma) &= \text{tgt}(\text{tgt}(A)\sigma) \\ \text{src}(C[A]) &= \text{src}(C[\text{src}(A)]) = \text{src}(C)[\text{src}(A)] \\ \text{tgt}(C[A]) &= \text{tgt}(C[\text{tgt}(A)]) \quad \square \end{aligned}$$

For co-initial proof terms A and B we can define partial operations *residual* ($/$), *join* (\sqcup), and *deletion* ($-$). The residual A / B is used to compute which redexes in A remain after contracting the redexes of B , $A \sqcup B$ is used to obtain a single proof term containing all redexes of A and B , and $A - B$ is used to delete the redexes of B from A . The binary *orthogonality* predicate (\perp) determines whether the redexes of two proof terms interfere with each other.

Definition 5.4. Let A and B be proof terms. The *orthogonality* predicate $A \perp B$, the *join* operation $A \sqcup B$, the *residual* operation A / B , and the *deletion* operation $A - B$ are inductively defined by the clauses in Table 1. We call A and B *orthogonal* if $A \perp B$.

The tool ProTeM⁴ [14] offers support for automatically calculating these operations and the examples in this paper were obtained with its help. Note that ProTeM is an experimental tool independent of the Isabelle formalization. It was developed to help better understand proof terms by implementing operations on proof terms like the ones described

⁴<http://informatik-protem.uibk.ac.at/>

in [27] and [14]. The main goal of ProTeM is to provide an intuitive and convenient user interface for quickly doing computations on examples, such as the ones in this paper.

Example 5.5. Consider again the TRS \mathcal{R}_1 of Example 2.1. We have

$$\begin{aligned} \alpha(g(a), b) &\perp h(f(y, g(\delta))) \\ \alpha(g(a), a) &\not\perp h(f(g(a), y)) \\ \alpha(g(a), b) \sqcup h(f(y, g(\delta))) &= \alpha(y, \delta) \\ \alpha(g(a), \delta) / h(f(y, g(\delta))) &= \alpha(g(b), a) \\ \alpha(y, \delta) - h(f(y, g(\delta))) &= \alpha(g(a), b) \\ \alpha(g(a), a) \star h(f(g(a), y)) &\text{ is undefined for } \star \in \{\sqcup, /, -\} \end{aligned}$$

We assume that $-$ and $/$ bind stronger than \sqcup . Note that \sqcup , $/$ and $-$ are partial operations. Hence the result of computing $A \sqcup B$, A / B , or $A - B$ is not always defined. Straightforward induction proofs on the definitions yield the following results.

Lemma 5.6. If A and B are orthogonal then A / B and $A \sqcup B$ are defined. \square

Lemma 5.7. If A / B and B / A are defined then

$$\text{src}(B / A) = \text{tgt}(A) \quad \text{and} \quad \text{tgt}(A / B) = \text{tgt}(B / A)$$

If $A \star B$ is defined then

$$\text{src}(A \star B) = \text{src}(A) = \text{src}(B)$$

for $\star \in \{\sqcup, -\}$. \square

Table 1. Operations on proof terms.

$$\begin{array}{l}
 x \perp x \quad x \sqcup x = x \quad x / x = x \quad x - x = x \\
 f(A_1, \dots, A_n) \perp f(B_1, \dots, B_n) \iff A_i \perp B_i \text{ for all } 1 \leq i \leq n \\
 \alpha(A_1, \dots, A_n) \perp \text{lhs}(\alpha)\langle B_1, \dots, B_n \rangle \iff A_i \perp B_i \text{ for all } 1 \leq i \leq n \\
 \text{lhs}(\alpha)\langle A_1, \dots, A_n \rangle \perp \alpha(B_1, \dots, B_n) \iff A_i \perp B_i \text{ for all } 1 \leq i \leq n \\
 f(A_1, \dots, A_n) \sqcup f(B_1, \dots, B_n) = f(A_1 \sqcup B_1, \dots, A_n \sqcup B_n) \\
 \alpha(A_1, \dots, A_n) \sqcup \alpha(B_1, \dots, B_n) = \alpha(A_1 \sqcup B_1, \dots, A_n \sqcup B_n) \\
 \alpha(A_1, \dots, A_n) \sqcup \text{lhs}(\alpha)\langle B_1, \dots, B_n \rangle = \alpha(A_1 \sqcup B_1, \dots, A_n \sqcup B_n) \\
 \text{lhs}(\alpha)\langle A_1, \dots, A_n \rangle \sqcup \alpha(B_1, \dots, B_n) = \alpha(A_1 \sqcup B_1, \dots, A_n \sqcup B_n) \\
 f(A_1, \dots, A_n) / f(B_1, \dots, B_n) = f(A_1 / B_1, \dots, A_n / B_n) \\
 \alpha(A_1, \dots, A_n) / \alpha(B_1, \dots, B_n) = \text{rhs}(\alpha)\langle A_1 / B_1, \dots, A_n / B_n \rangle \\
 \alpha(A_1, \dots, A_n) / \text{lhs}(\alpha)\langle B_1, \dots, B_n \rangle = \alpha(A_1 / B_1, \dots, A_n / B_n) \\
 \text{lhs}(\alpha)\langle A_1, \dots, A_n \rangle / \alpha(B_1, \dots, B_n) = \text{rhs}(\alpha)\langle A_1 / B_1, \dots, A_n / B_n \rangle \\
 f(A_1, \dots, A_n) - f(B_1, \dots, B_n) = f(A_1 - B_1, \dots, A_n - B_n) \\
 \alpha(A_1, \dots, A_n) - \alpha(B_1, \dots, B_n) = \text{lhs}(\alpha)\langle A_1 - B_1, \dots, A_n - B_n \rangle \\
 \alpha(A_1, \dots, A_n) - \text{lhs}(\alpha)\langle B_1, \dots, B_n \rangle = \alpha(A_1 - B_1, \dots, A_n - B_n)
 \end{array}$$

The properties below can be used to compute joins, residuals, and deletions if the proof terms involved adhere to certain patterns.

Lemma 5.8. Let $\star \in \{\sqcup, /, -\}$.

1. $A \star \text{src}(A) = A$
2. If $A \star B = D$ then $C[A] \star \text{src}(C)[B] = C[D]$ for any proof term context C .
3. If $\sigma(x) = \text{src}(\tau(x))$ for all $x \in \mathcal{V}\text{ar}(A)$ then $A\sigma \sqcup \text{src}(A)\tau = A\tau$.

Overlapping Proof Terms. We present an inductive definition for measuring the amount of overlap between co-initial proof terms. It is based on a special labeling of the source of a proof term. The following definition was introduced in [14].

Definition 5.9. We write $\text{lhs}^\#(\alpha)$ for the result of labeling every function symbol in $\text{lhs}(\alpha)$ with α as well as the distance to the root of α :

$$\text{lhs}^\#(\alpha) = \varphi(\text{lhs}(\alpha), \alpha, 0)$$

with $\varphi(t, \alpha, i) = t$ if $t \in \mathcal{V}$ and

$$\varphi(t, \alpha, i) = f_{\alpha^i}(\varphi(t_1, \alpha, i+1), \dots, \varphi(t_n, \alpha, i+1))$$

if $t = f(t_1, \dots, t_n)$. The mapping $\text{src}^\#$ computes the labeled source of a proof term: $\text{src}^\#(A) =$

$$\begin{cases}
 A & \text{if } A \in \mathcal{V} \\
 f(\text{src}^\#(A_1), \dots, \text{src}^\#(A_n)) & \text{if } A = f(A_1, \dots, A_n) \\
 \text{lhs}^\#(\alpha)\langle \text{src}^\#(A_1), \dots, \text{src}^\#(A_n) \rangle_\alpha & \text{if } A = \alpha(A_1, \dots, A_n)
 \end{cases}$$

The function ℓ extracts labels from function symbols: $\ell(f_{\alpha^n}) = \alpha^n$. The set of labeled positions for a proof term A is defined as

$$\mathcal{P}\text{os}_L(A) = \{p \in \mathcal{P}\text{os}(\text{src}^\#(A)) \mid \ell(\text{src}^\#(A)(p)) \text{ is defined}\}$$

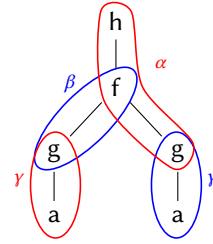


Figure 4. The redexes of A (red) and B (blue).

Example 5.10. Consider again the TRS \mathcal{R}_1 and the proof terms A and B of Example 5.1. A graphical representation of the redexes of A and B is depicted in Figure 4. We have

$$\text{src}^\#(A) = h_{\alpha^0}(f_{\alpha^1}(g_{\alpha^0}(a_{\alpha^1}), g_{\alpha^2}(a)))$$

$$\ell(\text{src}^\#(A)(1)) = \ell(f_{\alpha^1}) = \alpha^1$$

$$\mathcal{P}\text{os}_L(A) = \{\epsilon, 1, 11, 12, 111\}$$

For the proof term B we have

$$\text{src}^\#(B) = h(f_{\beta^0}(g_{\beta^1}(a), g_{\gamma^0}(a_{\gamma^1})))$$

$$\mathcal{P}\text{os}_L(B) = \{1, 11, 12, 121\}$$

Note that $\ell(\text{src}^\#(B)(\epsilon))$ and $\ell(h)$ are undefined.

By definition every label α^n occurring at position p in a labeled source must be nested below n function symbols which are labeled $\alpha^{n-1}, \dots, \alpha^0$.

Lemma 5.11. If $\ell(\text{src}^\#(A)(p)) = \alpha^n$ then $n \leq |p|$ and for all $q \in \mathcal{P}\text{os}_{\mathcal{F}}(\text{src}^\#(A))$ with $q \leq p$ and $|p \setminus q| \leq n$ we have

$$\ell(\text{src}^\#(A)(q)) = \alpha^{n-|p \setminus q|} \quad \checkmark$$

Corollary 5.12. For all $p \in \mathcal{P}\text{os}_L(A)$ with $\ell(\text{src}^\#(A)(p)) = \alpha^n$ there exists a position $q \in \mathcal{P}\text{os}_L(A)$ such that $|p \setminus q| = n$ and $\ell(\text{src}^\#(A)(q)) = \alpha^0$. \square

Lemma 5.13. *Given a proof term A and a position $p \in \text{src}(A)$ such that $\ell(\text{src}^\#(A)(p)) = \alpha^0$, there exists a unique position $p_\alpha \in \mathcal{P}\text{os}(A)$ such that $A = A[\alpha(\dots)]_{p_\alpha}$ and $\text{src}(A)[\]_p = \text{src}(A[\]_{p_\alpha})$. \checkmark*

Considering the labeled positions in $\text{src}^\#(A)$ we can split the proof term A into a term t without any rule symbols and a substitution ρ mapping the variables of t to proof terms (possibly containing rule symbols) such that $A = t\rho$.

Lemma 5.14. *Let $\text{src}(A) = t\sigma$ for some linear term t with variables x_1, \dots, x_n at positions p_1, \dots, p_n and some substitution σ . If $p \in \mathcal{P}\text{os}_L(A)$ for no $p \in \mathcal{P}\text{os}_\mathcal{F}(t)$ then $A = t\rho$ with $\rho = \{x_i \mapsto A|_{p_i} \mid 1 \leq i \leq n\}$. \checkmark*

Definition 5.15. For co-initial proof terms A and B we use the number of positions that are labeled in both $\text{src}^\#(A)$ and $\text{src}^\#(B)$ as a measure for the amount of overlap between A and B :

$$\blacktriangle(A, B) = |\mathcal{P}\text{os}_L(A) \cap \mathcal{P}\text{os}_L(B)|$$

Lemma 5.16. *For co-initial proof terms A and B*

$$A \perp B \iff \blacktriangle(A, B) = 0 \quad \square$$

To extract an innermost overlap, first we collect all pairs of overlapping redexes in co-initial proof terms.

Definition 5.17. For co-initial proof terms A and B the set $\text{overlaps}(A, B)$ consists of all pairs (p, q) of function symbol positions in the common source s of A and B such that

- (i) $\ell(\text{src}^\#(A)(p)) = \alpha^0$, $\ell(\text{src}^\#(B)(q)) = \beta^0$, and
- (ii) either $p \leq q$ and $\ell(\text{src}^\#(A)(q)) = \alpha^{|q \setminus p|}$ or $q < p$ and $\ell(\text{src}^\#(B)(p)) = \beta^{|p \setminus q|}$

or some rule symbols α and β .

The condition $\ell(\text{src}^\#(A)(q)) = \alpha^{|q \setminus p|}$ in the first case of the above definition ensures that $q \setminus p$ is a position in $\text{lhs}(\alpha)$ and likewise for the second case.

Lemma 5.18. *For co-initial proof terms A and B*

$$\text{overlaps}(A, B) = \emptyset \iff \blacktriangle(A, B) = 0 \quad \square$$

Definition 5.19. With each overlap $(p, q) \in \text{overlaps}(A, B)$ where $\ell(\text{src}^\#(A)(p)) = \alpha^0$ and $\ell(\text{src}^\#(B)(q)) = \beta^0$ we associate proof terms Δ_1 and Δ_2 corresponding to the two overlapping redexes in the common source s of A and B :

$$\begin{aligned} \Delta_1 &:= s[\alpha(s_1, \dots, s_n)]_p \text{ where } \text{lhs}(\alpha)\langle s_1, \dots, s_n \rangle_\alpha = s|_p \\ \Delta_2 &:= s[\beta(t_1, \dots, t_m)]_q \text{ where } \text{lhs}(\beta)\langle t_1, \dots, t_m \rangle_\beta = s|_q \end{aligned}$$

For left-linear rules α and β with $\text{vpos}(\alpha) = (p_1, \dots, p_n)$ and $\text{vpos}(\beta) = (q_1, \dots, q_m)$ we can write this more concisely as

$$\begin{aligned} \Delta_1 &= s[\alpha(s|_{pp_1}, \dots, s|_{pp_n})]_p \\ \Delta_2 &= s[\beta(s|_{qq_1}, \dots, s|_{qq_m})]_q \end{aligned}$$

Next we define the notion of innermost overlap.

Definition 5.20. We define the following order on overlaps:

$$(p_1, q_1) \leq (p_2, q_2) \iff p_1 \leq p_2 \text{ and } q_1 \leq q_2$$

An *innermost overlap* of co-initial proof terms A and B is a maximal element in $\text{overlaps}(A, B)$ with respect to \leq .

Example 5.21. Consider again the proof terms A and B of Example 5.1. From computing $\mathcal{P}\text{os}_L(A)$ and $\mathcal{P}\text{os}_L(B)$ in Example 5.10 we see that the source terms have overlap at positions 1, 11, and 12. Hence $\blacktriangle(A, B) = 3$. Computing the overlaps explicitly results in

$$\text{overlaps}(A, B) = \{(\epsilon, 1), (\epsilon, 12), (11, 1), \}$$

This situation is depicted in Figure 4. Both $(11, 1)$ and $(\epsilon, 12)$ are innermost overlaps. For $(11, 1)$ we have

$$\Delta_1 = h(f(\gamma, g(a))) \quad \Delta_2 = h(\beta(a, g(a)))$$

Note that it is not always the case that $\blacktriangle(A, B)$ is the same as $|\text{overlaps}(A, B)|$, since the same overlap can contribute to several overlapping positions in the source. Only the relation of Lemma 5.18 and the inequality $\blacktriangle(A, B) \geq |\text{overlaps}(A, B)|$ hold in general.

Remark. There is more than one sensible definition of *innermost* overlap. We also considered taking the following ordering on overlaps instead of the one in Definition 5.20: $(p_1, q_1) \leq (p_2, q_2)$ if either $\min(p_1, q_1) < \min(p_2, q_2)$, or $\min(p_1, q_1) = \min(p_2, q_2)$ and $\max(p_1, q_1) < \max(p_2, q_2)$. Using that alternative ordering, only the overlap $(\epsilon, 12)$ of Example 5.21 would be innermost. Both orders are suitable for proving Theorem 4.2 later on and we tried both options in the formalization. In the end we chose Definition 5.20 since it is slightly simpler.

6 Unification of Linear Variable-Disjoint Terms

In order to extract the critical pair underlying an overlap (p, q) between proof terms A and B , a most general unifier (mgu) of the left-hand sides of the involved rules needs to be computed. Due to left-linearity of the underlying TRS this is relatively straightforward since computing an mgu for linear terms that do not share variables is an especially easy instance of the unification problem.

Definition 6.1. For linear terms s and t without common variables, let $\text{vpos}(s) = (p_1, \dots, p_n)$, $\text{var}(s) = (x_1, \dots, x_n)$, $\text{vpos}(t) = (q_1, \dots, q_m)$, and $\text{var}(t) = (y_1, \dots, y_m)$. The substitution $\tau(s, t)$ is defined as follows:

$$\begin{aligned} \tau(s, t) &= \{x_i \mapsto t|_{p_i} \mid 1 \leq i \leq n \text{ and } p_i \in \mathcal{P}\text{os}(t)\} \\ &\cup \{y_j \mapsto s|_{q_j} \mid 1 \leq j \leq m \text{ and } q_j \in \mathcal{P}\text{os}_\mathcal{F}(s)\} \end{aligned}$$

For a set of equations $E = \{s_1 \approx t_1, \dots, s_k \approx t_k\}$ with linear and pairwise variable-disjoint terms, we denote the union $\tau(s_1, t_1) \cup \dots \cup \tau(s_k, t_k)$ by $\tau(E)$.

Lemma 6.2. *If s and t are unifiable and linear terms without common variables then $\tau(s, t)$ is an idempotent mgu of s and t .* ✓

Proof. Let $E = \{s_1 \approx t_1, \dots, s_k \approx t_k\}$ with linear and pairwise variable-disjoint terms $s_1, \dots, s_k, t_1, \dots, t_k$ and let S be a set of variable bindings $\{z_i \mapsto u_1, \dots, z_l \mapsto u_l\}$ such that

$$\{z_1, \dots, z_l\} \cap \mathcal{V}\text{ar}(s_1, \dots, s_k, t_1, \dots, t_k) = \emptyset$$

We show by induction over the definition of the unification algorithm [5, Chapter 4]⁵ that a derivation

$$E, S \implies \dots \implies \emptyset, S'$$

results in a set of variable bindings of the shape

$$S' = S \cup \tau(E)$$

The base case of the induction is $E = \emptyset$ in which case we immediately obtain $S' = S = S \cup \tau(\emptyset)$ as result. There are three step cases to consider from the unification algorithm:

Decompose. For a decomposition step

$$\begin{aligned} & \{f(s'_1, \dots, s'_n) \approx f(t'_1, \dots, t'_n)\} \uplus E', S \\ & \implies \{s'_1 \approx t'_1, \dots, s'_n \approx t'_n\} \cup E', S \end{aligned}$$

the induction hypothesis yields

$$S' = S \cup \bigcup_{i=1}^n \tau(s'_i, t'_i) \cup \tau(E')$$

Since

$$\bigcup_{i=1}^n \tau(s'_i, t'_i) = \tau(f(s'_1, \dots, s'_n), f(t'_1, \dots, t'_n))$$

we obtain the desired $S' = S \cup \tau(E)$.

Eliminate (left to right). For an equation $x \approx t \in E$ we know that $x \neq t$ by assumption. Consider the variable elimination step

$$\{x \approx t\} \uplus E', S \implies E' \sigma, S \sigma \cup \{x \mapsto t\}$$

where $\sigma = \{x \mapsto t\}$. We have $E' \sigma = E'$ and $S \sigma \cup \{x \mapsto t\} = S \cup \{x \mapsto t\}$ since all variables appearing in E' and S are distinct from x by assumption. Moreover the induction hypothesis yields $S' = S \cup \{x \mapsto t\} \cup \tau(E')$. The definition of τ ensures the equality $S \cup \{x \mapsto t\} \cup \tau(E') = S \cup \tau(E)$.

Eliminate (right to left). For an equation $s \approx y \in E$ we again know $s \neq y$ by assumption. The variable elimination step

$$\{s \approx y\} \uplus E, S \implies E' \sigma, S \sigma \cup \{y \mapsto s\}$$

with $\sigma = \{y \mapsto s\}$ is treated analogously to the preceding case. Here it is important to note that we may assume that s is not a variable. Hence the condition $q_j \in \mathcal{P}\text{os}_{\mathcal{F}}(s)$ in the definition of τ takes effect. □

⁵The proof follows the algorithm in Figure 4.5 in [5], which served as a template for the existing implementation in IsaFoR described in [25].

Example 6.3. Consider the terms $s = f(x_1, g(x_2))$ and $t = f(g(y_1), y_2)$. Applying the unification algorithm yields

$$\begin{aligned} & \{s \approx t\}, \emptyset \\ & \implies \{x_1 \approx g(y_1), g(x_2) \approx y_2\}, \emptyset \quad (\text{Decomposition}) \\ & \implies \{g(x_2) \approx y_2\}, \{x_1 \mapsto g(y_1)\} \quad (\text{Eliminate}) \\ & \implies \emptyset, \{x_1 \mapsto g(y_1), y_2 \mapsto g(x_2)\} \quad (\text{Eliminate}) \end{aligned}$$

The substitution $\{x_1 \mapsto g(y_1), y_2 \mapsto g(x_2)\}$ is exactly $\tau(s, t)$ as defined in Definition 6.1.

7 Main Proof

In the following we first show how, for each overlap, we can extract the corresponding critical peak of the underlying TRS.

Definition 7.1. Given an overlap $o = (p, q) \in \text{overlaps}(A, B)$ with $q \leq p$, we let

- α and β be the rule symbols at positions p and q in $\text{src}(A)$ and $\text{src}(B)$ such that $\ell(\text{src}^\#(A)(p)) = \alpha^0$ and $\ell(\text{src}^\#(B)(q)) = \beta^0$,
- $s = \text{src}(A) = \text{src}(B)$,
- $q' = p \setminus q$,
- $\text{vpos}(\alpha) = (p_1, \dots, p_n)$ and $\text{var}(\alpha) = (x_1, \dots, x_n)$,
- $\text{vpos}(\beta) = (q_1, \dots, q_m)$ and $\text{var}(\beta) = (y_1, \dots, y_m)$
- $q_\beta \in \mathcal{P}\text{os}(B)$ be the position of β in B according to Lemma 5.13 such that $B = B[\beta(B_1, \dots, B_m)]_{q_\beta}$ and $\text{src}(B)[\]_q = \text{src}(B[\]_{q_\beta})$

and assume $\{x_1, \dots, x_n\} \cap \{y_1, \dots, y_m\} = \emptyset$ without loss of generality. We define the substitutions

$$\begin{aligned} \sigma_o &= \{x_i \mapsto s|_{pp_i} \mid 1 \leq i \leq n\} \cup \{y_j \mapsto s|_{qq_j} \mid 1 \leq j \leq m\} \\ \tau_o &= \tau(\text{lhs}(\alpha), \text{lhs}(\beta)|_{q'}) \end{aligned}$$

The substitution σ_o maps the variables of $\text{lhs}(\alpha)$ and $\text{lhs}(\beta)$ to subterms of s such that $\text{lhs}(\alpha)\sigma_o = s|_p$ and $\text{lhs}(\beta)\sigma_o = s|_q$ and τ_o is an mgu of $\text{lhs}(\alpha)$ and $\text{lhs}(\beta)|_{q'}$. The key properties of these substitutions are captured by the following two lemmata.

Lemma 7.2. *For every overlap $o = (p, q) \in \text{overlaps}(A, B)$ with $q \leq p$*

$$\begin{array}{ccc} & \text{lhs}(\beta)[\text{lhs}(\alpha)\tau_o]_{q'} = \text{lhs}(\beta)\tau_o & \\ & \swarrow \alpha \quad \searrow \beta & \\ & q' \quad \epsilon & \\ \text{lhs}(\beta)[\text{rhs}(\alpha)\tau_o]_{q'} & & \text{rhs}(\beta)\tau_o \end{array}$$

is a critical peak. ✓

Proof. Since α and β both operate on the same source term s we know that $\text{lhs}(\alpha)$ and $\text{lhs}(\beta)|_{q'}$ are unifiable; simply take as unifier the substitution σ_o . In addition both $\text{lhs}(\alpha)$ and $\text{lhs}(\beta)$ must be linear and hence Lemma 6.2 is applicable and yields τ_o as idempotent mgu of $\text{lhs}(\alpha)$ and $\text{lhs}(\beta)|_{q'}$. In

addition τ_o does not replace any variables of $\text{lhs}(\beta)$ that occur at positions above or parallel to q' . Hence

$$\text{lhs}(\beta)[\text{lhs}(\alpha)]_{q'}\tau_o = \text{lhs}(\beta)[\text{lhs}(\alpha)\tau_o]_{q'} \quad \square$$

Lemma 7.3. *For every overlap $o = (p, q) \in \text{overlaps}(A, B)$ with $q \leq p$ we have $x_i\tau_o\sigma_o = s|_{pp_i}$ for $1 \leq i \leq n$ and $y_j\tau_o\sigma_o = s|_{qq_j}$ for $1 \leq j \leq m$, and hence*

$$\begin{aligned} s &= s[\text{lhs}(\beta)\tau_o\sigma_o]_q = s[\text{lhs}(\alpha)\tau_o\sigma_o]_p \\ &= s[\text{lhs}(\beta)[\text{lhs}(\alpha)\tau_o]_{q'}\sigma_o]_q \quad \checkmark \end{aligned}$$

Proof. For $1 \leq i \leq n$ we have $x_i\tau_o\sigma_o = x_i\sigma_o = s|_{pp_i}$ when $q'p_i \notin \text{Pos}(\text{lhs}(\beta))$. If $q'p_i \in \text{Pos}(\text{lhs}(\beta))$ then $x_i\tau_o\sigma_o = \text{lhs}(\beta)|_{q'p_i}\sigma_o = s|_{qq'p_i} = s|_{pp_i}$ since $\text{lhs}(\beta)\sigma_o = s|_q$. An analogous inspection yields $y_j\tau_o\sigma_o = s|_{qq_j}$ whenever $q_j \setminus q' \notin \text{Pos}_{\mathcal{F}}(\text{lhs}(\alpha))$ and $y_j\tau_o\sigma_o = \text{lhs}(\alpha)|_{q_j \setminus q'}\sigma_o = s|_{p(q_j \setminus q')} = s|_{qq_j}$ whenever $q_j \setminus q' \in \text{Pos}_{\mathcal{F}}(\text{lhs}(\alpha))$ since $\text{lhs}(\alpha)\sigma_o = s|_p$. \square

Assuming that the given TRS is development closed, we know there exists a multi-step $\text{lhs}(\beta)[\text{rhs}(\alpha)\tau]_{q'} \twoheadrightarrow \text{rhs}(\beta)\tau$. In the following the proof term representation of such a multi-step will be called D' . Inserting D' into a proper context yields a proof term D witnessing the multi-step $\text{tgt}(\Delta_1) \twoheadrightarrow \text{tgt}(\Delta_2)$ ($t' \twoheadrightarrow u'$ in Figure 3). This is captured by the following result.

Lemma 7.4. *Let $o = (p, q) \in \text{overlaps}(A, B)$ where $q \leq p$ and define $D = s[D'\sigma_o]_q$ for a proof term D' witnessing $\text{lhs}(\beta)[\text{rhs}(\alpha)\tau_o]_{q'} \twoheadrightarrow \text{rhs}(\beta)\tau_o$. Then D witnesses the multi-step $\text{tgt}(\Delta_1) \twoheadrightarrow \text{tgt}(\Delta_2)$. \checkmark*

Proof. We have

$$\begin{aligned} \text{src}(D) &= \text{src}(s[D'\sigma_o]_q) = s[\text{src}(D')\sigma_o]_q \\ &= s[\text{lhs}(\beta)\tau_o[\text{rhs}(\alpha)\tau_o]_{q'}\sigma_o]_q \\ &= s[\text{lhs}(\beta)\tau_o\sigma_o[\text{rhs}(\alpha)\tau_o\sigma_o]_{q'}]_q \\ &= s[s|_q[\text{rhs}(\alpha)\tau_o\sigma_o]_{q'}]_q \\ &= s[\text{rhs}(\alpha)\langle s|_{pp_1}, \dots, s|_{pp_n} \rangle_\alpha]_p \quad (7.3) \\ &= \text{tgt}(\Delta_1) \end{aligned}$$

and

$$\begin{aligned} \text{tgt}(D) &= \text{tgt}(s[D'\sigma_o]_q) = s[\text{tgt}(D')\sigma_o]_q \\ &= s[\text{rhs}(\beta)\tau_o\sigma_o]_q \\ &= s[\text{rhs}(\beta)\langle s|_{qq_1}, \dots, s|_{qq_m} \rangle_\beta]_q \quad (7.3) \\ &= \text{tgt}(\Delta_2) \quad \square \end{aligned}$$

The proof term D can be used to construct the proof term $D \sqcup (B - \Delta_2) / \Delta_1$ witnessing the multi-step $\text{tgt}(\Delta_1) \twoheadrightarrow \text{tgt}(B)$ which corresponds to the arrow $t' \twoheadrightarrow u$ in Figure 3. To show that the constructed proof term is well-defined, we use another substitution ρ_o , which is similar to σ_o but maps to subterms of B while σ_o maps to the sources of these proof terms. Recall from Definition 7.1 that B has the shape $B = B[\beta(B_1, \dots, B_m)]_{q_\beta}$ for some proof terms B_1, \dots, B_m

and position q_β . The purpose of the substitution ρ_o defined below is to reconstruct the arguments of β , i.e., the proof terms B_1, \dots, B_m , in the intermediate proof term D' .

Definition 7.5. Given an overlap $o = (p, q) \in \text{overlaps}(A, B)$ with $q \leq p$, let $B_j = B|_{q_\beta j}$ for $1 \leq j \leq m$. We define the substitution

$$\begin{aligned} \rho_o &= \{y_j \mapsto B_j \mid 1 \leq j \leq m\} \\ &\cup \{x_i \mapsto \text{lhs}(\beta)\langle B_1, \dots, B_m \rangle_\beta|_{q'p_i} \mid 1 \leq i \leq n\} \end{aligned}$$

Lemma 7.6. *If $1 \leq j \leq m$ then $\tau_o(y_j)\rho_o = B_j$. \checkmark*

Proof. We distinguish two cases: $\tau_o(y_j) = y_j$ and $\tau_o(y_j) \neq y_j$. In the first case we immediately obtain $\tau_o(y_j)\rho_o = \rho_o(y_j) = B_j$ from the definition of ρ_o . For the second case we first verify that $\text{src}^\#(\text{lhs}(\beta)\langle B_1, \dots, B_m \rangle_\beta|_{q'})(p')$ is unlabeled for $p' \in \text{Pos}_{\mathcal{F}}(\text{lhs}(\alpha))$. This relies on the fact that having a labeled function symbol at such a position p' would contradict the assumption that (p, q) is an innermost overlap of A and B . Then it follows from the second part of the definition of ρ_o and Lemma 5.14 that

$$\text{lhs}(\alpha)\rho_o = \text{lhs}(\beta)\langle B_1, \dots, B_m \rangle_\beta|_{q'} \quad (*)$$

Subsequently we obtain

$$\begin{aligned} \tau_o(y_j)\rho_o &= \text{lhs}(\alpha)|_{q_j \setminus q'}\rho_o \quad (\text{definition of } \tau_o) \\ &= (\text{lhs}(\beta)\langle B_1, \dots, B_m \rangle_\beta|_{q'})|_{q_j \setminus q'} \quad (\text{by } *) \\ &= \text{lhs}(\beta)\langle B_1, \dots, B_m \rangle_\beta|_{q_j} \\ &= B_j \quad \square \end{aligned}$$

Lemma 7.7. *Let $o = (p, q) \in \text{overlaps}(A, B)$ be an innermost overlap where $q \leq p$ and define D as in Lemma 7.4. Then*

1. $D \sqcup (B - \Delta_2) / \Delta_1 = B[D'\rho_o]_{q_\beta}$ \checkmark
2. $D \sqcup (B - \Delta_2) / \Delta_1$ witnesses $\text{tgt}(\Delta_1) \twoheadrightarrow \text{tgt}(B)$ $\checkmark \checkmark$

Proof. Since $\text{src}(B|_{q_\beta}) = s|_q$, Lemma 5.8(2) is applicable to compute $B - \Delta_2$:

$$B - \Delta_2 = B[\text{lhs}(\beta)\langle B_1, \dots, B_m \rangle_\beta]_{q_\beta} \quad (5.8(2))$$

$$= B[\text{lhs}(\beta)\tau_o\rho_o]_{q_\beta} \quad (7.6)$$

$$= B[\text{lhs}(\beta)[\text{lhs}(\alpha)\tau_o]_{q'}\rho_o]_{q_\beta} \quad (7.2)$$

Since $\text{src}(B[\text{lhs}(\beta)[\]_{q'}\rho_o]_{q_\beta}) = s|_p$, another application of Lemma 5.8(2) yields

$$(B - \Delta_2) / \Delta_1 = B[\text{lhs}(\beta)[\text{rhs}(\alpha)\tau_o]_{q'}\rho_o]_{q_\beta}$$

From Lemma 5.8(3) we obtain

$$D'\sigma_o \sqcup \text{lhs}(\beta)[\text{rhs}(\alpha)\tau_o]_{q'}\rho_o = D'\rho_o$$

and since $D = s[D'\sigma_o]_q$ we can apply Lemma 5.8(2) (modulo symmetry of \sqcup) to obtain the desired

$$D \sqcup (B - \Delta_2) / \Delta_1 = B[D'\rho_o]_{q_\beta}$$

From Lemmata 5.7 and 7.4 we obtain

$$\text{src}((B - \Delta_2) / \Delta_1) = \text{tgt}(\Delta_1) = \text{src}(D)$$

and hence

$$\text{src}(D \sqcup (B - \Delta_2) / \Delta_1) = \text{tgt}(\Delta_1)$$

It remains to show $\text{tgt}(D \sqcup (B - \Delta_2) / \Delta_1) = \text{tgt}(B)$. We have

$$\begin{aligned} & \text{tgt}(B[D'\rho_o]_{q_\beta}) \\ &= \text{tgt}(B[\text{tgt}(\text{rhs}(\beta)\tau_o\rho_o)]_{q_\beta}) \end{aligned} \quad (5.3)$$

$$= \text{tgt}(B[\text{tgt}(\text{rhs}(\beta)\langle B_1, \dots, B_m \rangle_\beta)]_{q_\beta}) \quad (7.6)$$

$$= \text{tgt}(B[\beta(B_1, \dots, B_m)]_{q_\beta}) \quad (5.3)$$

$$= \text{tgt}(B) \quad \square$$

Lemma 7.8. *Let $o = (p, q) \in \text{overlaps}(A, B)$ be an innermost overlap where $q \leq p$ and define D as in Lemma 7.4. Then*

$$\blacktriangle(A / \Delta_1, D \sqcup (B - \Delta_2) / \Delta_1) < \blacktriangle(A, B) \quad \checkmark$$

Proof (sketch). The proof is structured in three main parts corresponding to the equations (1)–(3) below. We first show

$$\begin{aligned} & r \in \mathcal{P}\text{os}_L(A / \Delta_1) \cap \mathcal{P}\text{os}_L(B[D'\rho_o]_{q_\beta}) \\ & \implies r < p \vee r \parallel p \end{aligned} \quad (1)$$

The proof proceeds by assuming $p \leq r$ and deriving a contradiction. This requires a lengthy case analysis on the position r and was one of the main challenges of the formalization. The implication (1) immediately yields

$$p \notin \mathcal{P}\text{os}_L(A / \Delta_1) \cap \mathcal{P}\text{os}_L(B[D'\rho_o]_{q_\beta}) \quad (2)$$

The implication (1) can moreover be used to obtain

$$\begin{aligned} & r \in \mathcal{P}\text{os}_L(A / \Delta_1) \cap \mathcal{P}\text{os}_L(B[D'\rho_o]_{q_\beta}) \\ & \implies r \in \mathcal{P}\text{os}_L(A) \cap \mathcal{P}\text{os}_L(B) \end{aligned} \quad (3)$$

The first part $r \in \mathcal{P}\text{os}_L(A)$ follows immediately from (1). Showing $r \in \mathcal{P}\text{os}_L(B[D'\rho_o]_{q_\beta})$ requires another tedious case analysis on r —the most difficult part being the case where r occurs below q . Finally we conclude

$$\blacktriangle(A / \Delta_1, B[D'\rho_o]_{q_\beta}) < \blacktriangle(A, B)$$

since there is at least one overlap less on the left-hand side, i.e., the overlap at position p . \square

Example 7.9. We continue our running example and as in Example 5.21 select as innermost overlap $o = (p, q)$ between A and B the overlap (11, 1) involving rule symbols γ and β . Hence $q' = 1$ and

$$\sigma_o = \{x \mapsto a, y \mapsto g(a)\}$$

$$\tau_o = \{x \mapsto a\}$$

$$\rho_o = \{x \mapsto a, y \mapsto \gamma\}$$

We obtain the critical peak

$$f(g(b), y) \xrightarrow{\gamma} f(g(a), y) \xrightarrow{\beta} f(g(a), g(a))$$

which can be closed by a multi-step represented by the proof term $D' = \beta(\delta, y)$ and hence $D = h(\beta(\delta, g(a)))$. The position of rule symbol β in B is $q_\beta = 1$ and the proof term

$$D \sqcup (B - \Delta_2) / \Delta_1 = B[D'\rho_o]_{q_\beta} = h(\beta(\delta, \gamma))$$

witnesses the multi-step

$$\text{tgt}(\Delta_1) = h(f(g(b), g(a))) \twoheadrightarrow h(f(g(a), g(a))) = \text{tgt}(B)$$

We have

$$\text{src}^\#(A / \Delta_1) = h_{\alpha^0}(f_{\alpha^1}(g(b), g_{\alpha^2}(a)))$$

$$\text{src}^\#(B[D'\rho_o]_{q_\beta}) = h(f_{\beta^0}(g_{\beta^1}(b_{\delta^0}), g_{\gamma^0}(a_{\gamma^1})))$$

and hence

$$\blacktriangle(A / \Delta_1, B[D'\rho_o]_{q_\beta}) = 2 < 3 = \blacktriangle(A, B)$$

The following two examples illustrate the need for selecting an innermost overlap in the preceding lemmata. The first example shows that the residual $(B - \Delta_2) / \Delta_1$ is not always defined for overlaps that are not innermost. The second example shows that even if we could come up with a different way of constructing some closing step C for $\text{tgt}(\Delta_1) \twoheadrightarrow \text{tgt}(B)$ we may run into the problem of $\blacktriangle(A / \Delta_1, C)$ not being smaller than $\blacktriangle(A, B)$.

Example 7.10. We can observe the effect of selecting the non-innermost overlap $(\epsilon, 1)$ in our running example. Since we require $q \leq p$ for the selected overlap (p, q) we need to swap A and B in this case. Furthermore we need to rename the variables in rules α and β to make them distinct. So we consider the TRS

$$\alpha \quad h(f(x_1, g(x_2))) \rightarrow h(f(x_1, g(x_1))) \quad \gamma \quad g(a) \rightarrow g(b)$$

$$\beta \quad f(g(y_1), y_2) \rightarrow f(g(y_1), g(y_1)) \quad \delta \quad b \rightarrow a$$

and the proof terms

$$A = h(\beta(a, \gamma)) \quad B = \alpha(\gamma, a)$$

with non-innermost overlap $(1, \epsilon)$. We obtain

$$\Delta_1 = h(\beta(a, g(a)))$$

$$\Delta_2 = \alpha(g(a), a)$$

$$B - \Delta_2 = h(f(\gamma, g(a)))$$

The term $\text{lhs}(\beta)$ does not match $B - \Delta_2$ and hence $(B - \Delta_2) / \Delta_1$ is undefined. Moreover the substitution ρ_o is not well-defined and there does not exist any substitution ρ such that

$$\text{lhs}(\beta)\rho = \text{lhs}(\alpha)\langle \gamma, a \rangle_{\alpha|_{q'}} = f(\gamma, a)$$

as required by Lemma 7.6 since the rule symbol γ is in the way.

Example 7.11. The left-linear TRS \mathcal{R}_2 consisting of the rewrite rules

$$\alpha \quad f(x) \rightarrow g(x, x) \quad \beta \quad a \rightarrow b \quad \delta \quad b \rightarrow c$$

$$\gamma \quad a \rightarrow c \quad \epsilon \quad c \rightarrow b$$

is development closed since the only critical pair $b \approx c$ arising from the root overlap between β and γ can be closed in both directions by δ and ϵ respectively. Consider the proof terms $A = \alpha(\beta)$ and $B = \alpha(\gamma)$. We have $\text{overlaps}(A, B) = \{(\epsilon, \epsilon), (1, 1)\}$ and $\blacktriangle(A, B) = 2$. By selecting the first overlap—which is not innermost—we obtain $\Delta_1 = \Delta_2 = \alpha(a)$. Then

$D'g(x, x)$ and $D = g(a, a)$ and therefore $D \sqcup (B - \Delta_2) / \Delta_1 = g(\gamma, \gamma)$. Since $A / \Delta_1 = g(\beta, \beta)$ the amount of overlap $\blacktriangle(A / \Delta_1, D \sqcup (B - \Delta_2) / \Delta_1)$ is again 2 since β and γ have been duplicated by the application of α .

We are now ready to prove our main theorem.

Proof of Theorem 4.2. Assume $t \leftarrow s \rightarrow u$ and let A be a proof term representing $s \rightarrow t$ and let B be a proof term representing $s \rightarrow u$. We show $t \rightarrow v \leftarrow u$ for some term v by well-founded induction on the amount of overlap between A and B . If $\blacktriangle(A, B) = 0$ then $A \perp B$ (Lemmata 5.18 and 5.16) and hence A / B and B / A are well-defined (Lemma 5.6) and represent the multi-steps $t \rightarrow \text{tgt}(A/B)$ and $u \rightarrow \text{tgt}(B/A)$ respectively. Since $\text{tgt}(A/B) = \text{tgt}(B/A)$ by Lemma 5.7 this proves the base case of the induction. If $\blacktriangle(A, B) > 0$ then $\text{overlaps}(A, B) \neq \emptyset$ and we can select an innermost overlap (p, q) and assume without loss of generality that $q \leq p$. Then we can apply the construction of Lemma 7.7. Therefore we obtain proof terms A / Δ_1 and $D \sqcup (B - \Delta_2) / \Delta_1$ such that $\text{tgt}(A / \Delta_1) = \text{tgt}(A)$ and $\text{tgt}(D \sqcup (B - \Delta_2) / \Delta_1) = \text{tgt}(B)$. The induction hypothesis can now be applied since $\blacktriangle(A / \Delta_1, D \sqcup (B - \Delta_2) / \Delta_1) < \blacktriangle(A, B)$ by Lemma 7.8 and yields multi-steps $t \rightarrow v$ and $u \rightarrow v$ for some common term v . \square

8 Formalization Details

We have integrated our formalization into IsaFoR, the Isabelle Formalization of Rewriting. This is an Isabelle/HOL library containing many useful results about first-order terms, substitutions, and contexts as well as numerous abstract results and concrete techniques from the term rewriting literature. Our new contributions consist of formalizing the operations on proof terms and labeling of the source of a proof term, as well as proving that the construction described in Section 7 does indeed yield a formal proof of the correctness of the development closedness criterion for inferring confluence. Of course this also includes the numerous smaller lemmata introduced in the previous sections and therefore has led to an extension of IsaFoR by more than 10000 new lines of code, including 300 new facts and 30 new definitions.

We have extended IsaFoR with a subfolder (and corresponding session) `Proof_Terms` which consists of the following files:

Utils.thy This file contains small useful results that are used in the formalization but are not necessarily related to proof terms or the confluence proof. This involves for example lemmata about variables, positions, substitutions, and contexts. It also contains the proof of Lemma 6.2 about unification of linear terms.

Proof_Terms.thy This file contains the datatype for proof terms and the definition of a notion of well-defined proof terms over a given TRS. We provide more details about these definitions below. The file also contains

definitions of the functions `src` and `tgt` and the proof of the correspondence between multi-steps and proof terms.

Residual_Join_Deletion.thy As the name suggests, this file contains definitions of the three partial operations `/`, `\sqcup` and `-` on proof terms. It also contains proofs that these operations fulfill Lemma 5.8 as well as various other properties of these operations, like for example the symmetry of `\sqcup`.

Orthogonal_PT.thy This file contains the definition of the orthogonality predicate (\perp) and the proofs that A / B and $A \sqcup B$ are well-defined whenever $A \perp B$ (Lemma 5.6).

Labels_and_Overlaps.thy This is by far the largest of the five files in this folder. Here we first define the type of a labeled term and the function that computes the labeled source of a proof term. Then we prove several properties of $\text{src}^\#(A)$ (e.g. Lemma 5.11). Finally the file also contains the definition of the measure function $\blacktriangle(A, B)$ and the function $\text{overlaps}(A, B)$ as well as properties of these functions and the connections $\text{overlaps}(A, B) = \emptyset \iff \text{measure}(A, B) = 0 \iff A \perp B$ (Lemmata 5.16 and 5.18).

In addition we added the file `Development_Closed.thy` inside `Confluence_and_Completion`. That file contains the proof of Theorem 4.2 as outlined in the previous section.

In the following we discuss some interesting details of the formalization. Proof terms are defined as an extension of first order terms:

type_synonym

```
(('f, 'v) pterm = (('f, 'v) prule + 'f, 'v) term"
```

Since proof terms are now essentially defined as terms, where function symbols are either rewrite rules of type `('f, 'v) prule` or the usual function symbols of type `'f`, we can reuse all existing results about first-order terms. This is especially useful when dealing with substitutions and contexts of proof terms. To increase readability when dealing with instances of this sum type, which usually requires writing `Inl` and `Inr` for the two options, we introduced the following abbreviations:

abbreviation Prule

```
where "Prule  $\alpha$  As  $\equiv$  Fun (Inl  $\alpha$ ) As"
```

abbreviation Pfun

```
where "Pfun f As  $\equiv$  Fun (Inr f) As"
```

We use the predicate `wf_pterm R` to check whether the correct number of arguments is provided for rule symbols and whether all rule symbols belong to a certain TRS R .

inductive_set wf_pterm for R where

```
"Var x  $\in$  wf_pterm R"
```

```
| " $\forall t \in \text{set } ts. t \in \text{wf\_pterm } R \implies$ 
```

```
  Pfun f ts  $\in$  wf_pterm R"
```

```
| "(lhs  $\alpha$ , rhs  $\alpha$ )  $\in$  R  $\implies$ 
```

```
  length As = length (var_rule  $\alpha$ )  $\implies$ 
```

```
   $\forall a \in \text{set } As. a \in \text{wf\_pterm } R \implies$ 
```

```
  Prule  $\alpha$  As  $\in$  wf_pterm R"
```

9 Integration into CeTA

The tool CeTA [20] is a certifier for, among other properties, (non-)confluence of rewrite systems with and without conditions. It can check certificates for confluence in CPF (certification problem format) [26] as generated by tools like CSI [17] and ACP [4]. All techniques verified by CeTA are proved in IsaFoR. To be precise, IsaFoR contains executable “check”-functions for each formalized proof technique together with proofs that whenever such a check is accepted, the technique is applied correctly and yields the correct result. CeTA is then obtained via code-generation directly from the Isabelle formalization. A graphical representation of the relation between IsaFoR and CeTA is shown in Figure 5. In order to implement the development closedness technique in CeTA we had to perform the following steps:

1. Provide a formal proof of Theorem 4.2 in Isabelle/HOL by extending the existing IsaFoR library.
2. Decide on an appropriate extension of the CPF and extend the parser for CeTA accordingly.
3. Provide an executable check function in IsaFoR that checks the conditions for development closed TRSs.

The most difficult step was proving Theorem 4.2 in IsaFoR. Our approach to the formalization has been extensively described in the previous sections. For extending the CPF we looked at the existing implementation of the parallel closedness condition [19] and modeled development closedness in an analogous fashion. This means that a CPF proof for development closedness simply consists of the given TRS together with the claim that it is development closed. Since CeTA together with its parser is obtained via code generation from IsaFoR, changing the parser to accept such certificates means changing the corresponding IsaFoR code. When CeTA reads a certificate it needs to check whether the claims in it are correct. For this purpose we provided an executable function in IsaFoR that takes as input a TRS \mathcal{R} and verifies all conditions of development closedness. So it first checks whether the TRS fulfills the variable conditions that $\text{lhs}(\alpha)$ is not a variable and $\text{Var}(\text{rhs}(\alpha)) \subseteq \text{Var}(\text{lhs}(\alpha))$ for all $\alpha \in \mathcal{R}$. Then it checks whether \mathcal{R} is left-linear. Finally the function computes all critical pairs $s \approx t$ of \mathcal{R} and checks whether there exists a multi-step $s \twoheadrightarrow_{\mathcal{R}} t$ for each of them. A function for computing all critical pairs of \mathcal{R} already existed in IsaFoR since this is also needed for many other confluence results based on critical pair criteria. An executable function for checking whether there exists a multi-step $s \twoheadrightarrow_{\mathcal{R}} t$ between arbitrary terms s and t was not yet present and had to be implemented by us.

To test this new extension of CeTA we used the confluence tool CSI [17] where the development closedness criterion was already implemented. Only a small adaptation of CSI’s certificate output was necessary⁶ in order to certify its output. We

⁶Per default CSI always applied the more general *almost* development closed criterion. Since CeTA does not yet incorporate this extension, we had to

then tested CeTA together with CSI on the COPS database.⁷ We did not observe a significant increase in the number of certifiable confluence proofs on this database. This is likely due to the already powerful certified strategy implemented in CSI (e.g., incorporating redundant rules). We expect however that some extensions of the development closedness criterion (as mentioned in the next section) may yield more certifiable proofs in the future.

10 Conclusion and Future Work

The development-closed criterion for left-linear TRSs by van Oostrom [30] is easy to state. It is also easy to automate and several confluence tools support it. It is not easy to formalize in a proof assistant. In this paper we presented the first formalization in Isabelle/HOL. The formalized proof is roughly based on the picture proof in [29, 30] but several obstacles had to be overcome. We obtained several formalized results on proof terms, which may benefit future formalizations (e.g. [31]).

There are a number of extensions of the development-closed criterion for confluence which are worthwhile to formalize since they will reduce the gap between the number of YES/NO answers by confluence tools and the number of certified YES/NO answers on COPS problems. At the time of writing, confluence tools deliver 503 YES/NO answers⁸ for the 562 TRSs in COPS, of which 375 are verified by CETA.⁹ The extensions include

- the *almost* development-closed criterion [29], in which the joinability requirement of critical peaks stemming from overlapping rules at the root is relaxed,
- commutation by relative termination [10, Theorems 3.2 and 4.3], in which (certain) critical steps are required to be terminating with regard to the original rewrite rules,
- hot-decreasing as well as critical pair closing systems [11, Theorems 2 and 3].

Especially the latter one is expected to reduce the certification gap significantly. Extending the formalization to higher-order pattern rewrite systems will be a formidable task since IsaFoR offers no support for higher-order terms.

Acknowledgments

This research was funded in part by the Austrian Science Fund (FWF) project I5943. The authors would like to thank Julian Nagele for starting the formalization of proof terms back in 2014 and René Thiemann for valuable input concerning Isabelle, especially on how to deal with conditional

differentiate between development closed and almost development closed systems.

⁷<http://cops.uibk.ac.at/>

⁸<http://cops.uibk.ac.at/results/?y=2021-full-run&c=TRS>

⁹<http://cops.uibk.ac.at/results/?y=2021-full-run&c=CPF-TRS>

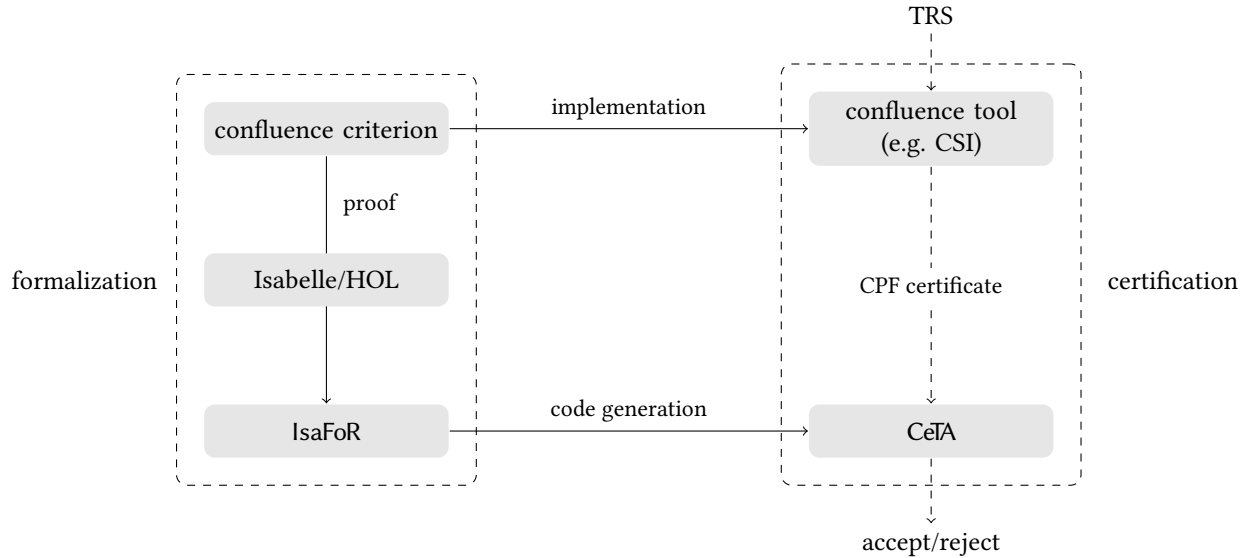


Figure 5. Schematic of the interaction between IsaFoR, confluence tools, and CeTA. Adapted from [18].

code equations. The valuable feedback of the anonymous reviewers improved the presentation.

References

- [1] Takahito Aoto. 2010. Automated Confluence Proof by Decreasing Diagrams based on Rule-Labeling. In *Proc. 21st International Conference on Rewriting Techniques and Applications (LIPICs, Vol. 6)*, Christopher Lynch (Ed.), 7–16. <https://doi.org/10.4230/LIPICs.RTA.2010.7>
- [2] Takahito Aoto. 2013. Disproving Confluence of Term Rewriting Systems by Interpretation and Ordering. In *Proc. 9th International Symposium on Frontiers of Combining Systems (LNCS, Vol. 8152)*, Pascal Fontaine, Christophe Ringeissen, and Renate A. Schmidt (Eds.), 311–326. https://doi.org/10.1007/978-3-642-40885-4_22
- [3] Takahito Aoto and Yoshihito Toyama. 2012. A Reduction-Preserving Completion for Proving Confluence of Non-Terminating Term Rewriting Systems. *Logical Methods in Computer Science* 8, 1 (2012), 1–29. [https://doi.org/10.2168/LMCS-8\(1:31\)2012](https://doi.org/10.2168/LMCS-8(1:31)2012)
- [4] Takahito Aoto, Junichi Yoshida, and Yoshihito Toyama. 2009. Proving Confluence of Term Rewriting Systems Automatically. In *Proc. 20th International Conference on Rewriting Techniques and Applications (LNCS, Vol. 5595)*, Ralf Treinen (Ed.), 93–102. https://doi.org/10.1007/978-3-642-02348-4_7
- [5] Franz Baader and Tobias Nipkow. 1998. *Term Rewriting and All That*. Cambridge University Press. <https://doi.org/10.1017/CBO9781139172752>
- [6] Alonzo Church and John Barkley Rosser. 1936. Some Properties of Conversion. *Trans. Amer. Math. Soc.* 39, 3 (1936), 472–482. <https://doi.org/10.1090/S0002-9947-1936-1501858-0>
- [7] Bertram Felgenhauer and René Thiemann. 2017. Reachability, Confluence, and Termination Analysis with State-Compatible Automata. *Information and Computation* 253, 3 (2017), 467–483. <https://doi.org/10.1016/j.ic.2016.06.011>
- [8] André Luiz Galdino and Mauricio Ayala-Rincón. 2010. A Formalization of the Knuth-Bendix(-Huet) Critical Pair Theorem. *Journal of Automated Reasoning* 45, 3 (2010), 301–325. <https://doi.org/10.1007/s10817-010-9165-2>
- [9] Nao Hirokawa and Aart Middeldorp. 2011. Decreasing Diagrams and Relative Termination. *Journal of Automated Reasoning* 47, 4 (2011), 481–501. <https://doi.org/10.1007/s10817-011-9238-x>
- [10] Nao Hirokawa and Aart Middeldorp. 2013. Commutation via Relative Termination. In *Proc. 2nd International Workshop on Confluence*, Nao Hirokawa and Vincent van Oostrom (Eds.), 29–33. Available from <http://cl-informatik.uibk.ac.at/iwc/iwc2013.pdf>.
- [11] Nao Hirokawa, Julian Nagele, Vincent van Oostrom, and Michio Oyamaguchi. 2019. Confluence by Critical Pair Analysis Revisited. In *Proc. 27th International Conference on Automated Deduction (LNAI, Vol. 11716)*, Pascal Fontaine (Ed.), 319–336. https://doi.org/10.1007/978-3-030-29436-6_19
- [12] Gérard Huet. 1980. Confluent Reductions: Abstract Properties and Applications to Term Rewriting Systems. *J. ACM* 27, 4 (1980), 797–821. <https://doi.org/10.1145/322217.322230>
- [13] Donald E. Knuth and Peter B. Bendix. 1970. Simple Word Problems in Universal Algebras. In *Computational Problems in Abstract Algebra*, John Leech (Ed.), Pergamon Press, 263–297.
- [14] Christina Kohl and Aart Middeldorp. 2018. ProTeM: A Proof Term Manipulator (System Description). In *Proc. 3rd International Conference on Formal Structures for Computation and Deduction (LIPICs, Vol. 108)*, Hélène Kirchner (Ed.), 31:1–31:8. <https://doi.org/10.4230/LIPICs.FSCD.2018.31>
- [15] Aart Middeldorp, Julian Nagele, and Kiraku Shintani. 2021. CoCo 2019: Report on the Eighth Confluence Competition. *International Journal on Software Tools for Technology Transfer* 23, 6 (2021), 905–916. <https://doi.org/10.1007/s10009-021-00620-4>
- [16] Julian Nagele, Bertram Felgenhauer, and Aart Middeldorp. 2015. Improving Automatic Confluence Analysis of Rewrite Systems by Redundant Rules. In *Proc. 26th International Conference on Rewriting Techniques and Applications (LIPICs, Vol. 36)*, Maribel Fernández (Ed.), 257–268. <https://doi.org/10.4230/LIPICs.RTA.2015.257>
- [17] Julian Nagele, Bertram Felgenhauer, and Aart Middeldorp. 2017. CSI: New Evidence – A Progress Report. In *Proc. 26th International Conference on Automated Deduction (LNAI, Vol. 10395)*, Leonardo de Moura (Ed.), 385–397. https://doi.org/10.1007/978-3-319-63046-5_24
- [18] Julian Nagele, Bertram Felgenhauer, and Harald Zankl. 2017. Certifying Confluence Proofs via Relative Termination and Rule Labeling. *Logical Methods in Computer Science* 13, 2:4 (2017), 1–27. [https://doi.org/10.23638/LMCS-13\(2:4\)2017](https://doi.org/10.23638/LMCS-13(2:4)2017)

- [19] Julian Nagele and Aart Middeldorp. 2016. Certification of Classical Confluence Results for Left-Linear Term Rewrite Systems. In *Proc. 7th International Joint Conference on Automated Reasoning (LNCS, Vol. 9807)*, Jasmin Christian Blanchette and Stephan Merz (Eds.), 290–306. https://doi.org/10.1007/978-3-319-43144-4_18
- [20] Julian Nagele and René Thiemann. 2014. Certification of Confluence Proofs using CeTA. In *Proc. 3rd International Workshop on Confluence*, Takahito Aoto and Delia Kesner (Eds.), 19–23. Available from <http://c-informatik.uibk.ac.at/iwc/iwc2014.pdf>.
- [21] Julian Nagele and Harald Zankl. 2015. Certified Rule Labeling. In *Proc. 26th International Conference on Rewriting Techniques and Applications (LIPICs, Vol. 36)*, Maribel Fernández (Ed.), 269–284. <https://doi.org/10.4230/LIPICs.RTA.2015.269>
- [22] Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. 2002. *Isabelle/HOL – A Proof Assistant for Higher-Order Logic*. LNCS, Vol. 2283. Springer. <https://doi.org/10.1007/3-540-45949-9>
- [23] Ana Cristina Rocha-Oliveira, André Luiz Galdino, and Mauricio Ayala-Rincón. 2017. Confluence of Orthogonal Term Rewriting Systems in the Prototype Verification System. *Journal of Automated Reasoning* 58, 2 (2017), 231–251. <https://doi.org/10.1007/s10817-016-9376-2>
- [24] José-Luis Ruiz-Reina, José-Antonio Alonso, María-José Hidalgo, and Francisco-Jesús Martín-Mateos. 2002. Formal Proofs About Rewriting Using ACL2. *Annals of Mathematics and Artificial Intelligence* 36, 3 (2002), 239–262. <https://doi.org/10.1023/A:1016003314081>
- [25] Christian Sternagel and René Thiemann. 2013. Formalizing Knuth–Bendix Orders and Knuth–Bendix Completion. In *Proc. 24th International Conference on Rewriting Techniques and Applications (LIPICs, Vol. 21)*, Femke van Raamsdonk (Ed.), 287–302. <https://doi.org/10.4230/LIPICs.RTA.2013.287>
- [26] Christian Sternagel and René Thiemann. 2014. The Certification Problem Format. In *Proceedings of the 11th International Workshop on User Interfaces for Theorem Provers (Electronic Proceedings in Theoretical Computer Science, Vol. 167)*, Christoph Benzmüller and Bruno Woltzenlogel Paleo (Eds.), 61–72. <https://doi.org/10.4204/EPTCS.167.8>
- [27] TeReSe (Ed.). 2003. *Term Rewriting Systems*. Cambridge Tracts in Theoretical Computer Science, Vol. 55. Cambridge University Press.
- [28] Yoshihito Toyama. 1988. Commutativity of Term Rewriting Systems. In *Programming of Future Generation Computers II*, Kazuhiro Fuchi and Laurent Kott (Eds.). North-Holland, 393–407.
- [29] Vincent van Oostrom. 1995. Development Closed Critical Pairs. In *Proc. 2nd International Workshop on Higher-Order Algebra, Logic, and Term Rewriting (LNCS, Vol. 1074)*, Gilles Dowek, Jan Heering, Karl Meinke, and Bernhard Möller (Eds.), 185–200. https://doi.org/10.1007/3-540-61254-8_26
- [30] Vincent van Oostrom. 1997. Developing Developments. *Theoretical Computer Science* 175, 1 (1997), 159–181. [https://doi.org/10.1016/S0304-3975\(96\)00173-9](https://doi.org/10.1016/S0304-3975(96)00173-9)
- [31] Vincent van Oostrom and Roel de Vrijer. 2002. Four Equivalent Equivalences of Reductions. In *Proc. 2nd International Workshop on Reduction Strategies in Rewriting and Programming (Electronic Notes in Theoretical Computer Science, Vol. 70(6))*, Bernhard Gramlich and Salvador Lucas (Eds.), 21–61. [https://doi.org/10.1016/S1571-0661\(04\)80599-1](https://doi.org/10.1016/S1571-0661(04)80599-1)

Received 2022-09-21; accepted 2022-11-21